LIS - SUMMARY

FS17

Simone Guggiari

1. SUPERVISED LEARNING

classifier: data → label
train model with labeled data: $X \to Y$, $f: X \to Y$

Data representation
label y:

	binary/cat	ordinal	quantitative
	int: regression/bool	int: order	order & value matter

$x \in \mathbb{R}^d$ feature vectors

Linear regression

$y_i \approx f(x_i) = w \cdot x_i + b$

$= \sum_{j=1}^{d} w^{(j)} x^{(j)} + b = w^T x_i + b = \tilde{w}^T \tilde{x}$

Homogeneous: $\tilde{x} = [x, 1]$, $\tilde{w} = [w, b]$

$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(d)} \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(d)} \end{bmatrix} \in \mathbb{R}^{n \times d}$, $Y \in \mathbb{R}^n$

accuracy: $\hat{R}(\omega) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$ → ML Alg: $w^* = \operatorname{argmin}_w \hat{R}(w)$

error

closed form: take derivative, set it to 0

$\hat{R}(w) = (Y - Xw)^T (Y - Xw) = \|Y - Xw\|_2^2$

$\nabla_w \hat{R}(w)|_{w=w^*} = -X^T Y - X^T Y + 2X^T X w^* = 0 \rightsquigarrow$

$w^* = (X^T X)^{-1} X^T y$

derivative solution: obj fct is convex

Gradient descent (η_t = learning rate)

small step length → GD converges to w^* in finite # steps

$\nabla \hat{R}(w) = \left[\frac{\partial \hat{R}}{\partial w^{(1)}} \dots \frac{\partial \hat{R}}{\partial w^{(d)}} \right]^T$

$\frac{\partial \hat{R}}{\partial w^{(j)}} = \frac{\partial}{\partial w^{(j)}} \left[\sum_{i=1}^{n} (y_i - w^T x_i)^2 \right] = \sum_{i=1}^{n} \frac{\partial}{\partial w^{(j)}} (y_i - w^T x_i)^2 = \sum_{i=1}^{n} 2(y_i - w^T x_i)(-x_i^{(j)})$

$\rightarrow \nabla_w \hat{R}(w) = -2 \sum_{i=1}^{n} (y_i - w^T x_i) x_i = -2X^T (Y - Xw)$

GD($\vec{w}_0, \hat{R}(\cdot)$)
$t = 0$
while $\|\hat{R}(w_t)\| > TOL$
  $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$
  $t = t + 1$
return w_t

→ or for $t = 1, 2, \dots$

Adaptive step size η_t

line search: minimize the residual along direct gradient $\eta_t = \operatorname{argmin}_\eta \hat{R}(w_t - \eta \nabla \hat{R}(w_t))$

bold driver heuristic (speed η_t): $\hat{R}(w_{t+1}) < \hat{R}(w_t) \Rightarrow \eta_{t+1} = c_+ \eta_t$ (increase), $\Rightarrow c_- \eta_t$ (decr) $c_+ > 1 > c_-$

closed form: $O(d^3)$

iterative sol: $O(\#\text{it} \cdot nd)$, $\#\text{it} = \#\text{iterations}$

other loss fct

$\ell_1(w) = |y_i - w^T x_i|$ → robust against outliers

$\ell_2 = (\quad)^2$ → allows to define closed form

ℓ_p ($p \geq 1$, p real) → the bigger p the bigger outliers penalize

loss: $\hat{R}(w) = \sum_{i=1}^{n} \ell(w, x_i, y_i)$ → can use other functions

Model selection $D_{train} \rightarrow w^* = \operatorname{argmin}_w \hat{R}_{train}(w)$

we want w to be min on all data, not only train (have rest $\hat{R}$)

$(x,y) \sim P(X,Y) \rightarrow R(w) = \int_{x,y} P(x,y) (y - w^T x)^2 dx dy = \mathbb{E}_{x,y}[(y - w^T x)^2]$

unknown $G = \mathbb{E}_{x,y}[\ell(w; x, y)]$ loss $\ell(w; x, y)$

$\mathbb{E}_w[R_{train}(w)] \leq \mathbb{E}_w[R(w)]$ in general, model selection performed st.

⚠ never optimize parameters over train D as it will lead to overfit = non-robust model, ie wrong predictions

evaluation

∀ model (k): repeat for $i = 1 \dots m$ (m = # disjoint subsets)

split: $D = D_{train}^{(i)} \cup D_{test}^{(i)}$, $D_{train}^{(i)} \cap D_{test}^{(i)} = \emptyset$

train: $w_k^{(i)} = \operatorname{argmin}_{w \in \mathbb{R}^d} R_{D_{train}^{(i)}}(w)$

performance: $R_k^{(i)} = \hat{R}_{D_{test}^{(i)}}(w_k^{(i)})$ average: $\hat{R}_k = \frac{1}{m} \sum R_k^{(i)}$

select best model $k^* = \operatorname{argmin}_k \hat{R}_k$

splitting:

bootstrap: pick valid. set of given size randomly, rest is training

m-fold: train on $(m-1)$ folds, test on remaining

usually $m \in [5 \dots 10]$, m maximal → overfit test set, underfit train set

leave one out: use $m = n$-fold (LOOCV) comp. intensive

Regularization only try to minimize loss → model complexity ↑

avoid overfit → regularize term

Ridge regression: $R(\omega) = \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$ ← regularization term

closed form: $w^* = (X^T X + \lambda I)^{-1} X^T y$ ← inverse always invertible (since SPD)

gradient desc: $\nabla_w R(w) = \nabla_w \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \nabla_w \lambda \|w\|_2^2$

solve as lin. reg. $= \lambda \nabla_w w^T w = 2\lambda w$

Linear classification

labels discrete values (not real) = classes eg. binary classification

$y_i \in \{-1, +1\}$ $f(x) = w^T x$ → on the line = 0

$y_i = \text{sign}(w^T x_i) = \text{sign}(\cos \alpha)$

find linear separator: $w^* = \operatorname{argmin}_w \hat{R}(w) = \operatorname{argmin}_w \sum_{i=1}^{n} \ell(w; x_i, y_i)$

0/1 loss: counts # misclassific $\ell_{0/1}(w; x_i, y_i) = \begin{cases} 0 & \text{if } \text{sign}(w^T x_i) = y_i \\ 1 & \text{else} \end{cases}$ → NP hard / no convex diff. continuous

many poss. sol

perceptron loss: surrogate fct, convex, cont

$\ell_P(w; x_i, y_i) = \begin{cases} 0 & \text{if } \text{sign}(w^T x_i) = y_i \\ -y_i \cdot w^T x_i & \text{else} \end{cases}$ distance to dividing line if wrong

can use GD $= \max\{0, -y_i \cdot w^T x_i\}$

$\nabla_w R(w) = \nabla_w \sum \ell(w, x_i, y_i) = \nabla_w \sum \max\{0, -y_i \cdot w^T x_i\} = \sum_{i=1}^{n} \nabla_w \max\{0, -y_i \cdot w^T x_i\}$

$\frac{\partial}{\partial w^{(j)}} \ell_P = \begin{cases} 0 & \text{if } y_i \cdot w^T x_i \geq 0 \\ -y_i x_i^{(j)} & \text{else} \end{cases}$ $\nabla_w \hat{R}(w) = -\sum \begin{cases} 0 & \text{if } y_i \cdot w^T x_i \geq 0 \\ y_i x_i & \text{else} \end{cases}$

Stochastic GD to avoid summation over all n gradients

chooses at line •

$w_{t+1} = w_t - \eta_t \nabla_w \ell(\vec{w}; x_i, y_i)$ (i.u.a.r.)

Support Vector Machines (SVM) lin/lp don't guarantee unique sol → max margin of correct classific (robust against noise)

hinge loss $\ell_h(w; x_i, y_i) = \max\{0, 1 - y_i \cdot w^T x_i\}$

$w^* = \operatorname{argmin}_w \sum_{i=1}^{n} \ell_h(w; x_i, y_i) + \lambda \|w\|_2^2$ → reg. term necessary

⚠ without reg, $\|w\|_2 \to \infty$ st $y_i w^T x_i > 1$ → desired margin (unit)

(scaling w allows data pt to be in margin without loss)

use SGD or derivative $\frac{\partial}{\partial w^{(j)}} \ell_h + \frac{\partial}{\partial w^{(j)}} \lambda \|w\|_2^2 = \begin{cases} 0 & \text{if } 1 - y_i w^T x_i \leq 0 \\ -y_i x_i^{(j)} & \text{else} \end{cases} + 2\lambda w^{(j)}$

Non-linear predictions transform input with non-lin fct

polynomial: $f(x) = \sum_{i=1}^{n} w_i \phi_i(x)$, $\phi_i(x) = x^i$ [poly expansion $\vec{x} = [x_1, x_2]$

more info up to x^k → $\mathcal{O}(d^k)$ $\phi(\vec{x}) = [x_1, x_2, x_1^2, x_2^2, x_1 x_2, x_1^3, \dots]$

implicit approach: w^* lives in span of data $w^* = \sum_{i=1}^{n} (\alpha_i y_i) \vec{x}_i$ | SGD: $w_{t+1} = w_t + \eta_t y_i x_i$

revisiting perceptron

$w^* = \operatorname{argmin}_{w \in \mathbb{R}^n} \sum_{i=1}^{n} \max\{0, -y_i (\sum_{j=1}^{n} \alpha_j y_j x_j)^T x_i\} = \operatorname{argmin}_{w \in \mathbb{R}^n} \sum_{i=1}^{n} \max\{0, -y_i \sum_{j=1}^{n} \alpha_j y_j (x_j^T x_i)\}$

$\hookrightarrow \sum_{i=1}^{n} \max\{0, -y_i \sum_{j=1}^{n} \alpha_j y_j k(x_j, x_i)\}$

eg. fitting quadratic $f(x) = ax^2 + bx + c$ $\sum_{i=1}^{n} w_i \phi_i(x)$ $\phi_i(x) = x^i$, $a, b, c \in \mathbb{R}$

$\phi(x) = x \dots$

2 too high order of poly → not robust

best model

train

error = poly degree

$x \mapsto \phi(x)$

$x^T x' \mapsto \phi^T(x) \phi(x) = k(x, x')$

Kernels: fct that applied to two data points efficiently computes inner product in possibly non lin transf. space

es: 2 dim, deg 2: $x = [x_1, x_2]$ $\phi(x) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)$

$\phi^T(x) \phi(x') = \dots = (x_1 x_1' + x_2 x_2')^2 = (x^T x')^2$

don't comp in $\mathbb{R}^3$, but $\mathbb{R}^2$ & apply non-lin ∀ 2

$x \in \mathbb{R}^d \to \phi^T(x) \phi(x) \in \mathbb{R}^{O(d^p)}$

polynomial kernels

homogeneous degree p	$k(x, x') = (x^T x')^p$
all up to degree p	$k(x, x') = (1 + x^T x')^p$

kernelized perceptron

Perceptron ($\vec{w}_0$)	KernelPerceptron ($\vec{\alpha}_0$)
for t in $[1 \dots T]$	for t in $[1 \dots T]$
pick i uar	pick i uar
if $y_i \cdot w_{t-1}^T x_i > 0$	if $y_i \cdot \sum_{j=1}^{n} \alpha_{t-1}^{(j)} y_j k(x_i, x_j) > 0$
$w_t = w_{t-1}$	$\alpha_t = \alpha_{t-1}$
else	else
$w_t = w_{t-1} + \eta_t y_i x_i$	$\alpha_t^{(i)} = \alpha_{t-1}^{(i)} + \eta_t$, $\alpha_t^{(j)} = \alpha_{t-1}^{(j)}$ $\forall j \neq i$
return w_t	return α_t

implem are equivalent

kernelized SVM → $w_p = \operatorname{argmin}_{w \in \mathbb{R}^p} \sum_{i=1}^{n} \max\{0, 1 - y_i \sum_{j=1}^{n} w_j y_j k(x_i, x_j)\} + \lambda w^T K w$

$K \in \text{Gram matrix}$ (matrix of inner prod) | classify: $y(\vec{x}) = \text{sign}[\sum_{i=1}^{n} \alpha_i^* y_i k(x_i, \vec{x})]$

$K_{ij} = \langle x_i, x_j \rangle = x_i^T x_j$

kernel properties

must be mercer prod in suitable space $k: X \times X \to \mathbb{R}$ | All const ≥ 0 are kernels!

- symmetric: $k(x, x') = \phi^T(x) \phi(x') = \phi^T(x') \phi(x) = k(x', x)$
- positive definite (for semi?) > 0 is allowed in sem

Semi-pos matrix $M \in \mathbb{R}^{n \times n}$ symmetric

$x^T M x \geq 0 \quad \forall x \in \mathbb{R}^n \Rightarrow$ spectral positiveness (all $\in \mathbb{R}$ are non-neg) ⇔ positive eigenvalues

$M = U^T D U \Rightarrow x^T U^T D U x = (Ux)^T D (Ux) = \|Ux\|_2^2 D \geq 0$

(diagonalized form) $M u_i = \lambda_i u_i \to u_i^T M u_i = \lambda_i u_i^T u_i = \lambda_i \|u_i\|^2 \geq 0 \Rightarrow \lambda_i \geq 0$

Gram matrix $S = \{x_{i,n} x_{i,2}, \dots\} \in X$

$\phi: \mathbb{R}^d \to \mathbb{R}^D$ is valid kernel if corresponding Gram is psd

$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) \dots & k(x_1, x_n) \\ k(x_2, x_1) & \dots & k(x_2, x_n) \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix} = \Phi^T \Phi$, $\Phi = [\phi(x_1) \dots \phi(x_n)] \in \mathbb{R}^{D \times n}$

pos. semi-def $x^T K x = x^T \Phi^T \Phi x = v^T v \geq 0$

Kernels

linear = $x^T x'$

polyn = $(1 + x^T x')^n$

gauss = $\exp(-\|x - x'\|_2^2 / h^2)$

sigmoid = $\tanh(K x^T x' - b)$

engineering → combine existing ones

k_1, k_2 valid

$k = k_1 + k_2$

$= k_1 - k_2$

$= c \cdot k_1$ $c > 0$

$= f(k_1)$, exp or positive pol

Kernelized linear regression

w^* lies in span of data $w^* = \sum_{i=1}^{n} \alpha_i \vec{x}_i$

$\alpha^* = \operatorname{argmin}_\alpha \sum_{i=1}^{n} (\sum_{j=1}^{n} \alpha_j k(x_i, x_j) - y_i)^2 + [\text{reg}]$

$\|w\|_2^2 = w^T w = (\sum \alpha_i x_i)^T (\sum \alpha_j x_j) = \alpha^T K \alpha$

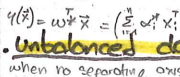
$\alpha^* = \operatorname{argmin}_\alpha \|K\alpha - y\|_2^2 + \lambda \alpha^T K \alpha \to \alpha^* = (K + \lambda I)^{-1} y$

$y(\vec{x}) = w^T x = (\sum \alpha_i x_i^T) x = \sum \alpha_i k(x_i, x)$ closed sol

Unbalanced classification take FP-FN

when no separating axis exists, counting misclassific → not optimal

balance the problem: • subsampling, • upsampling



repeat at random

repeat points from minority with perturb

cost-sensitive class

$w^* = \operatorname{argmin}_w \sum_{i:y_i=+1}^{n} c_+ \max\{0, -y_i w^T x_i\} + \sum_{i:y_i=-1}^{n} c_- \max\{0, -y_i w^T x_i\}$

$w^* = \operatorname{argmin}_w \sum_{i=1}^{n} \ell_{PC}(w; x_i, y_i)$ $\ell_{PC}(w; x_i, y_i) = c_{y_i} \max\{0, -y_i w^T x_i\}$

(perceptron) c_+, c_- weights error differently for different classes

if $\#\oplus < \#\ominus \Rightarrow c_+ > c_-$

Evaluating learning systems

confusion matrix

(⊕ is the rare class)

	Real +	Real −	
Pred +	TP	FP	p_+
Pred −	FN	TN	p_-
	n_+	n_-	n

metrics: Accuracy $\frac{TP+TN}{n} \in [0,1]$ — correctly classified

Precision $\frac{TP}{p_+}$ — how precisely a rare class is recognized

Recall $\frac{TP}{n_+}$ — the capacity of detecting the rare class

F1 score $= \frac{2}{\frac{1}{Prec} + \frac{1}{Rec}} = \dots = \frac{2TP}{2TP + FP + FN}$ — "goodness" used to optimize both Prec & Recall (inverse mean = harmonic mean)

	Rec	Rec	F1
w_1	low	high	max
w_2	med	med	
w_3	high	low	

Prec-rec curve: Prec ↑ by ↓ Rec, change some parameter

ROC curve: true positive rate $TPR = \frac{TP}{n_+}$ = recall; false positive rate $FPR = \frac{FP}{n_-}$

(best = 1) (best = 0)

Random classifier → 0.5 in both

TPR perf: sensitivity, specificity

parameter sweep → determine best classifier

$D = \{(x_i, y_i)\}_{i=1 \dots n}$, $y_i \in \{-1, +1\}$

Multiclass classification

One vs all → solve c bin classifiers, label is most confident classifier

$y_i = \operatorname{argmax}_{j \in [1 \dots c]} f_j(x_i)$ eg. $f_j(x) = w_j^T x / \|w_j\|_2$ (further away from line)

One vs one: train $\binom{c}{2} = \frac{c(c-1)}{2}$ classif for each pair (i,j), use voting scheme, y_i set to class most of the times predicted. always separable, OvA is not (except if clusters) OvO comp exp ($O(c^2)$)

Neural Networks feature engineering not always easy, must take relations into account

NN: find efficient features from data.

traditional pred: $w^* = \operatorname{argmin}_w \sum_{i=1}^{n} \ell(y_i, \sum_{j=1}^{n} w_j \phi_j(x_i))$ $\phi()$ does feature extraction

idea: extend learnable params, not only w, but $\nu, \theta \rightarrow$ feature functions

$w^* = \operatorname{argmin}_{w, \theta} \sum_{i=1}^{n} \ell(y_i, \sum_{j=1}^{n} w_j \phi(x_i; \theta_j))$

Activation fct $\phi(x, \theta) \equiv \varphi(\theta^T x)$

w, θ free, ϕ fixed

sigmoid $\varphi(z) = \frac{\lambda}{\lambda + e^{-z}}$

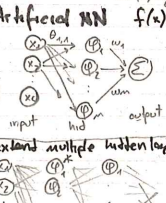
$\varphi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \tanh(z)$

$\max(0, z)$

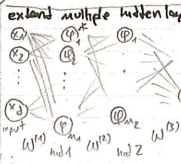
ReLU (rectified lin. unit) $= \max\{0, z\}$

Artificial NN $f(x) = \sum_{j=1}^{m} w_j \varphi(\theta_j^T x)$ → class of hypothesis to predict

$\theta = [\theta_1 \dots \theta_m]^T$ (raw)



extend multiple hidden layers & outputs = deep NN



chain of lin fct $\theta^T x$ followed by non lin scalar activation

input to $\varphi()$ is $z = \sum_j w_j^{(1)} x_j$

output of φ is input for next layer

Forward propagation

if we have optimized params, compute output for new $\vec{x}$ by traversing NN

FP $(\vec{x})$

foreach unit j in input layer: $v_j = x_j$ // initialize

foreach hidden layer $\ell = [1 \dots L-1]$

  foreach unit j in layer ℓ: $v_j = \varphi(\sum_{i \in \text{layer}_{\ell-1}} w_{j,i} v_i)$ // activ fct

foreach unit j in output layer: $f_j = \sum_{i \in \text{layer}_{L-1}} w_{j,i} v_i$ // output layer

REGR: predict f_j

CLASS: predict $\text{sign}(f_j)$

return y_i

matrix form

foreach x:

$\vec{v}^{(1)} = x$

foreach layer $\ell = [1 \dots L-1]$

$v^{(\ell)} = \varphi(W^{(\ell)} v^{(\ell-1)})$

for output layer

$f = W^{(L)} v^{(L-1)}$

$W^{(\ell)} = \begin{bmatrix} w_{1,1}^{(\ell)} & \dots \\ \vdots & w_{n_\ell, d}^{(\ell)} \end{bmatrix} \in \mathbb{R}^{n_\ell \times d}$

$\begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} ac \\ bd \end{bmatrix}$

Backward propagation

→ use b to determine weights

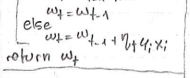
$D = \{(x_i, y_i)\}_{i=1 \dots n} \rightarrow$ find $W = (W^{(1)}, W^{(L)})$

measure goodness $\ell(W; x, y) = b(y - f(x, (W))) \overset{e.g.}{=} (y - f(x, (W)))^2$

regularize: restrict # layers & units $w^* = \operatorname{argmin}_w \sum_{i=1}^{n} \ell(W; x_i, y_i)$ → non convex

$w_{i \to j}$ links j to i | SGD: must compute $\nabla_W \ell(W, x, y)$

simple ex



f free params (w, w) $W = [w', w]$

$f(W, x) = w \cdot \varphi(z) = w \cdot \varphi(w' \cdot x)$ error signal

$\frac{\partial}{\partial w} \ell(y, f(x, W)) = \frac{\partial \ell}{\partial f} \frac{\partial}{\partial w} f(x, w) = \ell'(f) \cdot \varphi(w' \cdot x) = \delta v$ ← error signal

with squared loss $\ell = (y - f(x, W))^2$, $\delta = 2(f(x, W) - y)$

compute them using forward prop

$\frac{\partial}{\partial w'} \ell(y, f(x, W)) = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial v} \frac{\partial v}{\partial w'} = \delta \cdot w \cdot \varphi'(w' \cdot x) \cdot x$ ← comp using forward prop & φ'

δ' = error signal on hidden lev. $\delta' \to \delta$

Activ. fct derivatives

sigmoid	$\sigma'(z) = \sigma(z)(1 - \sigma(z))$	compactly supported
ReLU	$\varphi'(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$	

complex ex

$L(N) = \sum_i \ell_i \left(f_i \left(\sum_j w_{i,j} \varphi \left(\sum_k w'_{j,k} x_k \right) \right) \right)$

$\ell_i(f_i) = \text{quad}(y_i - f_i)^2$

$W = [w_{11}, \dots w_{32}, w'_{11} \dots w'_{32}]$

in w' hid w out

$\frac{\partial}{\partial w_{ij}} L = \frac{\partial \ell_i}{\partial w_{ij}} (w_{ij} \text{ only appears in } \ell_i) = \ell_i'(f_i) \cdot \varphi \left(\sum_{k=1}^{3} w'_{jk} x_k \right) = \delta_i v_j$

δ_i v_j

chain rule

$\frac{\partial}{\partial x} f(g(x)) = f'(g(x)) \cdot g'(x)$

$\frac{\partial}{\partial w'_{jk}} L = \frac{\partial}{\partial w'_{jk}} \sum_{i=1}^{2} \ell_i(\cdot) = \sum_{i=1}^{2} \frac{\partial}{\partial w'_{jk}} \ell_i(\cdot) = \sum_{i=1}^{2} \frac{\partial \ell_i}{\partial f_i} \frac{\partial f_i}{\partial v_j} \frac{\partial v_j}{\partial w_{jk}} =$

$= \sum_{i=1}^{2} \ell_i'(f_i) f_i'(v_j) v_j'(z_j) = \sum_{i=1}^{2} \delta_i w_{ij} \varphi'(z_j) x_k = \delta_j' x_k$ ← δ_j'

more complex → just sum over more index i (output) or k (input)

more hid → recursion (current input x_k as hidden input v_k, extend chain rule)

BackwardProp finds optimal weights w, θ

BP

foreach unit i in output layer: $\delta_i = \ell_i'(f_i)$ // error signal

foreach unit j in layer L: $\frac{\partial}{\partial w_{ij}} = \delta_i v_j$ // derivat between output and last hidd.

foreach unit j in hid. lay $\ell = [L-1 \dots 1]$

  $\delta_k = \varphi'(z_j) \sum_{i \in \text{layer}_{\ell+1}} w_{ij} \delta_i$

  foreach unit k in layer $\ell-1$: $\frac{\partial}{\partial w_{jk}} = \delta_j v_k$ // derivat between ℓ^{th} and $(\ell-1)^{th}$ layers

matrix form

output layer: $\delta^{(L)} = \ell'(f) = [\ell'(f_1) \cdot \ell'(f_2)]$

$\nabla_{W^{(L)}} \ell(w, x, y) = \delta^{(L)} v^{(L-1)}$ ← matrix of partial der. $\in \mathbb{R}^{p \times n_{L-1}}$

∀ hidd. $\ell = [L-1 \dots 1]$: $\delta^{(\ell)} = \varphi(z^{(\ell)}) \odot (W^{(\ell+1)})^T \delta^{(\ell+1)}$

$\nabla_{W^{(\ell)}} \ell(w, x, y) = \delta^{(\ell)} v^{(\ell-1)}$

$\odot$ = comp-wise mult.

⚠ non convex → local optima, depends on initialization of W

NN vs kernel opt

diff optimize params:

NN $f(x) = \sum w_i \tanh \theta_i^T x$

θ and w (non convex)

⊕ flexible → multiple layers = multiple lvls of abstr.

⊖ many data, tune params

KR $f(x) = \sum \alpha_i \tanh x_i^T x$

only w (convex)

⊕ robust against noise

⊖ grows with size of D

2. UNSUPERVISED LEARNING

No labels

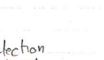
supervised classif/regress	unsupervised clustering	dim reduction

no validation step

Clustering find relations to group samples → outlier detection

hierarchical clustering: build tree — cut at certain level

partitional approach: define cost fct for partition & optimize it (es cut graph into disconnected components)



model based approaches: mixture cluster & infer membership

k-means clustering assign sample to closest center

$x \in$ cluster $i^* \Leftrightarrow i = \operatorname{argmin}_j \|x - \mu_j\|_2$, μ_j = center

norms: euclidean, manhattan

$L[\mu_1 \dots \mu_k] = \sum_{i=1}^{n} \min_{j \in [1 \dots k]} \|x_i - \mu_j\|_2^2 \to x_i$ $\mu^* = [\mu_1^*, \dots \mu_k^*] = \operatorname{argmin}_\mu L$

Lloyd's heuristic

$\mu^{(0)} = [\mu_1^{(0)} \dots \mu_k^{(0)}]$ // init

while not converged:

  foreach sample x_i: $z_i = \operatorname{argmin}_{j \in [1 \dots k]} \|x_i - \mu_j^{(t-1)}\|_2^2$ // cluster assignment

  foreach center μ_j: $\mu_j^{(t)} = \frac{1}{n_j} \sum_{i: z_i = j} x_i$ // cluster update

iterative, finds local min

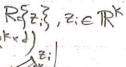
cost decreases monotone $L(\mu^{(t)}) \leq L(\mu^{(t-1)})$

iteration explosive

previous experiment $\mathcal{O}(nkd) + nd$

k must be known

only spherical shape



selecting k: cross-validate it by swapping different k

$k = n \to$ bsso introduce reg.

Dimension reduction $D = \{x_i\}$, $x_i \in \mathbb{R}^d \to$ find $R = \{z_i\}$, $z_i \in \mathbb{R}^k$

$f: \mathbb{R}^d \to \mathbb{R}^k$ $k \ll d$ les lin $R = AD$, $A \in \mathbb{R}^{k \times d}$

1D $[w^*, z_{1:n}^*] = \operatorname{argmin}_{\|w\|_2 = 1, z \in \mathbb{R}^n} \sum_{i=1}^{n} \|z_i w - x_i\|_2^2$

data must be centered $\sum_{i=1}^{n} x_i = 0$ $z_i = w^T x_i = \|w\| \|x_i\| \cos\alpha$

$w^* = \operatorname{argmax}_{\|w\|_2 = 1} \sum_{i=1}^{n} (w^T x_i)^2$ → finds the w that is max orthy aligned with data

$\Sigma(w^T x_i)^2 = \sum w^T x_i x_i^T w = w^T (\sum x_i x_i^T) w$

$d > 1$ $w^* = \operatorname{argmax}_{\|w\|_2 = 1} w^T \Sigma w$, $\Sigma = \frac{1}{n} X X^T = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$

PCA → eigenvector with largest EW of cov matrix Σ

centered data → ensure $\mu = \sum x_i = 0$

covariance estimate it with empirical cov $\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$

projection compute k principal EV of cov matrix $W = [w_1 | w_2 | \dots | w_k] \to z_i = W^T x_i$

they solve $\min \Sigma \|W z_i - x_i\|_2^2$

Non-linear dim red

use kernel trick

$x \in \mathbb{R}^d \mapsto \phi(x) \in \mathbb{R}^D \to z \in \mathbb{R}^k$ $\Sigma w^* = (X^T X) w^* = \lambda w^* = X^T (X w^*)$ $w^* = \sum_{i=1}^{n} \alpha_i x_i$

$w^* = \frac{1}{\lambda} X^T \beta = \frac{1}{\lambda} \sum_{i=1}^{n} \beta_i x_i$

choose $\alpha_i = \frac{\beta_i}{\lambda}$

$\Rightarrow$ span of data pts

Σ example $w^* = \operatorname{argmax}_{\|w\|_2 = 1} \sum_{i=1}^{n} (w^T x_i)^2 = \operatorname{argmax}_{w \in \mathbb{R}^n} \sum_{i=1}^{n} ((\sum_{j=1}^{n} w_j x_j)^T x_i)^2 = \operatorname{argmax}_{w \in \mathbb{R}^n} \sum_{i=1}^{n} (\sum_{j=1}^{n} w_j x_j^T x_i)^2$

kernels introduced → PCA works with non-linearities

$\alpha^* = \arg\max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^n \left(\sum_{j=1}^n \alpha_j k(x_i, x_j) \right)^2$ → matrix form: $\alpha^* = \arg\max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^n (\alpha^T K_{:i})^2$

reset normalization $\|w\|_2^2 = 1 \Rightarrow \alpha^T K \alpha = 1$ $\quad$ $= \alpha^T K^T K \alpha$

closed form sol: eigen decomp of kernel matrix K

$K = \sum_{i=1}^n \lambda_i v_i v_i^T \quad \lambda_1 \geq \lambda_2 \geq \dots \lambda_n \geq 0 \qquad \alpha^{*(i)} = \frac{1}{\sqrt{\lambda_i}} v_i$, v_i = ith principal EV of K

performing proj

given x, project onto $z \in \mathbb{R}^k$ using non-lin:

$z_i = w^{(i)T} \phi(\vec{x}) = \sum_{j=1}^n \alpha_j^{(i)} \phi(x_j)^T \phi(\vec{x}) = \sum_{j=1}^n \alpha_j^{(i)} k(x_j, \vec{x})$

3. STATISTICAL PERSPECTIVE ON SUP. LEARNING

investigate statistical assumptions placed on data

• Stat perspective on Regression

assume samples iid $(x_i, c_i) \sim P(X, Y)$ (unknown)

learn fct $h: \mathcal{X} \to \mathcal{Y}$ that minimizes pred. error $R(h) = \int P(x,y)\, l(y, h(x))\, dx\, dy = E_{xy}[\, l(y, h(x))]$

complete knowledge assume P known

$\min_h R(h) = \min_h E_x[E_y[(y - h(x))^2 | x]] = E_x[\min_{g(\vec{x})} E_y[\cdot]]$ → for choices of h, optimize over $g(\vec{x})$

$g(\hat{h}) = E_Y[(y - \hat{h})^2 | x]$ use deriv $\stackrel{!}{=} 0$

$\frac{d}{d\hat{h}} g(\hat{h}) = \frac{d}{d\hat{h}} \int (y - \hat{h})^2 P(y|x)\, dy = \int \frac{d}{d\hat{h}} (y - \hat{h})^2 P(y|x)\, dy \sim \int 2(\hat{h} - y) P(y|x)\, dy = 2\hat{h} \underbrace{\int P(y|x)\, dy}_{1} - 2 \underbrace{\int y P(y|x)\, dy}_{E[Y|x]} \stackrel{!}{=} 0$

$h^*(x) = E[Y|X=x]$ (logical) $\quad$ joint $P_{XY}(x,y)$

Bayes' optimal predictor, but supposes cond $P_{Y|X}(y|x)$

use $\hat{P}(Y|X)$, empirical based on samples $\quad \hat{y} = \hat{h}(x) = \hat{E}[Y|X=x] = \int y\, \hat{P}(y|X=x)\, dy$

MLE (maximum likelihood est) (explains regress)

estimate $\hat{P}(Y|X)$ by tuning params $\quad P(Y|X, \theta) \quad$ prior

$\theta^* = \arg\max_\theta \hat{P}(q_1 \dots q_n | x_1 \dots x_n, \theta) \overset{iid}{=} \arg\max_\theta \prod_{i=1}^n \hat{P}(q_i | x_i, \theta)$

$\theta^* = \arg\min_\theta - \sum_{i=1}^n \log \hat{P}(q_i | x_i, \theta)$ negative log-likelihood (min log of the above) | log is monotonic

→ choose form of param distr

ex: regr with gaussian distrib

$P(Y=y|X=x, \Theta) = \mathcal{N}(y; h(x), \sigma^2) = \mathcal{N}(y, w^T x, \sigma^2) \qquad Y = w^T X + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)$

$\theta = [w, \sigma^2]$ → determine minimizing log-likelihood

$L(w) = -\sum_{i=1}^n \log P(q_i | x_i, w, \sigma^2) = -\sum_{i=1}^n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(q_i - w^T x_i)^2}{2\sigma^2} \right)$

$= -n \log \frac{1}{\sqrt{2\pi\sigma^2}} - \mathcal{E} = \frac{(q_i - w^T x_i)^2}{\sigma^2} = \frac{n}{2} \log 2\pi\sigma^2 + \frac{1}{2\sigma^2} \sum_{i=1}^n (q_i - w^T x_i)^2$

$w^* = \arg\min_{w,\sigma} L(w) = \arg\min_{w,\sigma} \sum_{i=1}^n (q_i - w^T x_i)^2$ | → solving least squares → implicit gaussian assumption

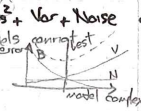
• Bias-Variance Tradeoff

$\text{PredError} = \text{Bias}^2 + \text{Var} + \text{Noise}$

bias: caused by model limitation, high when family of models cannot represent data. Even if $P(X,Y)$ then h^* is not perfect

Var: caused by having finite data

Noise: err in optimal model, error that cannot be reduced



MLE → unbiased if h^* can be represented by model class

↳ has minimum variance

overfit: introduce a bit of bias (reg) to reduce Var dramatically

Prior: constraint on weights $w \sim \mathcal{N}(0, \beta^2 \mathbb{I})$ $\quad$ prior $\quad$ likelihood

(explains regularize)

Bayes rule MAP $P(w | x_{1:n}, q_{1:n}) = \frac{P(w) \cdot P(q_{1:n} | x_{1:n}, w)}{P(q_{1:n} | x_{1:n})}$ posterior / evidence

$w^* = \arg\max_w P(w | x_{1:n}, q_{1:n}) = \arg\min_w - \log P(w) - \log P(q_{1:n} | x_{1:n}, w) + \log P(q_{1:n} | x_{1:n})$ | without prior = MLE

Bayes

prior $\quad$ const

Prior: $-\log P(w) = -\log \prod_{j=1}^d P(w_j) = -\sum_{j=1}^d \log P(w_j) = \dots = \frac{1}{2} \log 2\pi\beta^2 + \frac{1}{2\beta^2} \|w\|_2^2 + \text{const}$

Likelihood: $-\log P(q_{1:n} | x_{1:n}, w) = \frac{1}{2\sigma^2} \sum_{i=1}^n (q_i - w^T x_i)^2 + \text{const}$ $\quad$ mult by $2\sigma^2$

$\Rightarrow w^* = \arg\min_w (\dots + (\dots)) = \dots = \arg\min_w \lambda \|w\|_2^2 + \sum_{i=1}^n (q_i - w^T x_i)^2$

= Ridge regression (with regularizer)

↳ = MAP with gaussian prior on weights $\quad \Leftrightarrow \quad \lambda = \frac{\sigma^2}{\beta^2}$

Gauss: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ $\quad$ $\arg\max_w P(w) \prod_{i=1}^n P(q_i | x_i, w)$ or $P(w | x_{1:n}, x_{1:n})$

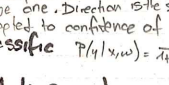
λ by: σ^2 huge (data really noisy) or β^2 small (believe w are small) $\quad$ $\frac{\sigma^2}{\beta^2}$ = Var → how noisy / var of weights

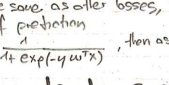
• General MAP estimation

$\arg\min_w \sum_{i=1}^n l(w^T x_i; x_i, q_i) + C(w) \Leftrightarrow \arg\max P(w|D)$ $\quad$ $C(w) = -\log P(w)$, $l(\cdot) = -\log P(q_i | x_i, w)$

ex l_2-reg → LASSO, provide sparsity in w → Laplace prior

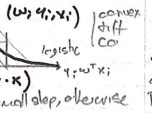
l_2-reg → gaussian prior on weights $\quad$ $w_i \sim \mathcal{L}(0, b) \Rightarrow P(w_i) = \frac{1}{2b} \exp - \frac{|w_i|}{b}$





• Stat. persp on classification

Logistic regression discrete labels → bernoulli distrib



sample far away from line → very confident

$P(q_i = 1 | w, x_i) = \sigma(d) = \sigma(w^T x)$

use dist $w^T x = d$ → sigmoid function $\quad \sigma(z) = \frac{1}{1 + e^{-z}}$ (bernoulli noise)

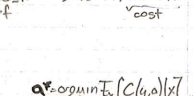
$P(q_i | x_i, w) = \frac{1}{1 + \exp - q_i \cdot w^T x} = \sigma(q_i \cdot w^T x)$

MLE separating hyperplane can be defined by maximizing likelihood of D

$\arg\max_w P(q_i | x_i, w) = \arg\min_w - \sum_i \log P(q_i | x_i, w)$

$-\log P = \dots = \log(1 + \exp(-q_i w^T x_i)) =: \ell_{\text{logistic}}(w, q_i, x_i)$ | convex, diff, cont

$w^* = \arg\min_w \sum_i \ell_{\text{logistic}}(w, x_i, y_i)$



$\nabla_w \ell_{\text{logistic}}(w) = \sigma(-w^T x)(-q_i w) = P(Y = -q_i | x_i, w) \cdot (-q_i \cdot x_i)$

computed using chain rule

SGD: if we are certain about classif, then perform small step, otherwise large one. Direction is the same as other losses, but step size is adapted to confidence of prediction

classific $P(q_i | x_i, w) = \frac{1}{1 + \exp(-q_i w^T x)}$, then assign to more probable class

• Acting under uncertainty

costs based on action

compute expected cost based on probabilities of spam/not & select min $\quad \mathbb{E}[\text{cost}]$

ex $p = P(q = S | x) = 0.2$

action	S	N
lost important mail	0	10
lost trash	N	1
	A	0.5	0.5

$S = \{0.2 \cdot 0 + 0.8 \cdot 10 = 8$
$N = 0.2 \cdot 1 + 0.8 \cdot 0 = 0.2$
$A = 0.2 \cdot 0.5 + 0.8 \cdot 0.5 = 0.5$

Bayesian decision theory

$P(Y=q|X)$ $\quad$ $A = \{a_0, a_1, \dots\}$ actions, Y outcomes, $C: Y \times A \to \mathbb{R}$ cost fct $\quad$ $a^* = \arg\min_a \mathbb{E}_Y[C(q,a)|x] = \arg\min_a \sum_y P(q=y|x) \cdot C(q,a)$

Symmetric costs ex binary log regression

$A = \{-1, +1\}$, $C(q,a) = \mathbb{I}_{a \neq q} = \begin{cases} 1 & a \neq q \\ 0 & \text{else} \end{cases}$

$\mathbb{E}_Y[C(q,a)] = \dots = P(q \neq a | x, w) = \sigma(-a \cdot w^T x)$

$a^* = \arg\min \mathbb{E}_Y[C(q,a)|x] = \arg\max_a (1 + \exp(a \cdot w^T x)) = \arg\max_a a \cdot w^T x$ $\quad \Rightarrow a^* = \text{sign}(w^T x)$

Asymmetric costs for $C(q,a)$

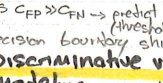
cost of action $\quad C_+ = \mathbb{E}[C(q,a=+1)|x] = 0 \cdot P(q=+1|x,w) + C_{FP} P(q=-1|x,w)$

$C_- \qquad\qquad -1 \qquad\qquad = C_{FN} \cdot \qquad\qquad + 0 \cdot$

predict $a=+1$ if $C_+ < C_-$, ie when $P(q=+1|x,w) > \frac{C_{FP}}{C_{FP} + C_{FN}}$ (threshold)

ex $C_{FP} \gg C_{FN}$ → predict $a=+1$ only when very confident

decision boundary shifted → wrt C_{FP}/C_{FN}



Act	cost $q=+1$	cost $q=-1$
+1	0	C_{FP}
-1	C_{FN}	0

4 Discriminative vs Generative modeling

previously modeled cond. prob

now model joint $P(Y,X)$

ex find outliers (logistic overconfident for outliers)

discriminative → conditional

generative → joint $\quad P(y|x) = \frac{P(y,x)}{P(x)} = \frac{P(y,x)}{\int_y P(x,y)\, dy}$

use prior $P(q)$ $\quad P(q|x) = \frac{1}{Z} P(q) P(x|q)$

ex spit dataset, prior on digits $= \frac{1}{10}$

• Naive-Bayes Model

$P(Y=q) = p_q$, $\sum_q p_q = 1$ $\quad$ independent features (all generative) $\quad P(x_{i:n} | q) = \prod_{i=1}^d P(x_i | q)$

Gaussian NB classifier assume features are gaussian distributed

$P(x_i | q) = \mathcal{N}(x_i | \mu_{q,i}, \sigma^2_{q,i})$ → does not feature dependent

use prior on Y, $P(Y_q)$ ppp then new $y = \{-1, +1\}$ $\quad$ use MLE

$P(D_q = [q_{1:n}] | p) = p^{n_+} \cdot (1-p)^{n_-}$ $\quad$ log-lik $\quad \ell(p) = \log(P(D_q|p)) = n_+ \log p + n_- \log(1-p)$

$\frac{d}{dp} \ell(p) = \frac{n_+}{p} + \frac{n_-}{1-p} \cdot (-1) \stackrel{!}{=} 0 \Rightarrow p^* = \frac{n_+}{n_+ + n_-}$

assume gaussian distrib → compute its params

$\mu_{q,i} = \frac{1}{n_q} \sum_{x \in D_{x|q}} x_i \qquad n_q = |D_{x|y}| \qquad D_{x|q} = \{x \in \mathbb{R}^n | y_i = x_i, q_i = q\}$

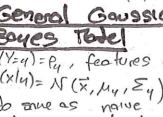
$\sigma^2_{q,i} = \frac{1}{n_q} \sum_{x \in D_{x|q}} (x_i - \mu_{q,i})^2$ $\quad$ **PREDICTION**

we now have $P(q_i)$, $P(x_i|q) \to P(q|x)$ of $P(q_i) P(x_i|q)$

$q^* = \arg\max_q P(q|x) = \arg\max_q P(q) P(x|q) \overset{\text{NBC}}{=} \arg\max_q P(q) \prod_{i=1}^n P(x_i|q)$

$= \arg\max \log(\sim) = \arg\max \log P(q_i) + \sum_{i=1}^n \log P(x_i|q_i)$

Decision boundary 1D only 1 feature, prior $P(y=\pm 1) = 0.5$



$\Rightarrow$ max prob = defining decision boundaries $= \prod P(x_i|y)$

class. rule: $\hat{y} = P(Y|x) = \frac{P(y)\, P(x|y)}{P(x)}$

Restrictions many features x_i are dependent → no previous assumpt. (x_1, x_4 correlated)

• General Gaussian Bayes Model

$P(Y=q) = p_q$, features are multivariate gaussian $\quad P(x|q) = \mathcal{N}(\vec{x}, \mu_q, \Sigma_q)$ → Naive used $\Sigma = \begin{bmatrix} \sigma^2_{q,1} & & c \\ & \ddots & \\ c & & \sigma^2_{q,d} \end{bmatrix}$

to save as naive ▲

need now to estimate entire multivariate gaussian

$\mu_q^* = \frac{1}{n_q} \sum_{x_i : q_i = q} \vec{x}_i \qquad n_q = |\{x_i : q_i = q\}|$

$\Sigma_q^* = \frac{1}{n_q} \sum_{x_i : q_i = q} (x_i - \mu_q^*)(x_i - \mu_q^*)^T$

Outlier detection find outlier by computing their prob

$P(x_{outlier}) = \sum_q P(x_{outlier}, q) = \sum_q P(q) \cdot P(x_{outlier} | q) \leq T_{outlier}$

don't pass threshold → outlier

↳ prob of being normal

• Dealing with missing data

unsup. learning → only labels missing

$D = \begin{bmatrix} x_1^{(1)} & \dots & N/A & & x_n^{(1)} & q_1 \\ x_1^{(2)} & & & & & q_2 \\ N/A & & & & N/A & \\ x_1^{(n)} & & & & & q_n \end{bmatrix}$

different gaussians $P(X|Y)$

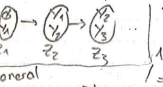
$P(X=x|Y=q) = \mathcal{N}(x, \vec{\mu}_q, \Sigma_q)$

gaussian mixtures $\quad P(x) = \int_y P(x,y)\, dy = \int_y P(y) P(x|y)\, dy$

$P(x) = \sum_{q=1}^k p_q \mathcal{N}(x, \vec{\mu}_q, \Sigma_q)$ $\quad$ discrete $= \sum_{q=1}^k P(q) P(x|q)$

convex linear comb of gaussians

$P(x, \theta) = P(x | \mu, \Sigma, \omega) = \sum_{i=1}^k \omega_i \mathcal{N}(x; \mu_i, \Sigma_i) \qquad \theta = (\mu, \Sigma, \omega)$



Fitting $\theta^* = \{\mu^*, \Sigma^*, \omega^*\} = \arg\min - \sum_{i=1}^n \log \sum_{j=1}^k \omega_j \mathcal{N}(x_i | \mu_j, \Sigma_j)$

non convex, constraints $\sum \omega = 1$, Σ spd

Hard expectation maximization (EM) const weights $\omega_j^{(0)} = \frac{1}{k}$

- init all params $\theta^{(0)}$ (ex cov matrix $\Sigma_j^{(0)}$, const weights)
- iterate until convers
  - $z_i := \arg\max_z P(z | x_i, \theta^{(t-1)}) = \arg\max_z P(z|\theta^{(t-1)}) P(x_i | z, \theta^{(t-1)})$ (predict most likely class with current params) $\quad \omega_z^{(t-1)} \cdot \mathcal{N}(x_i; \mu_z^{(t-1)}, \Sigma_z^{(t-1)})$
  - labels have been generated, D is now complete
  - update params for GB classifier

Soft EM to avoid overconfidence, since HARD EM assigning fixed label

↳ computes posterior prob over clusters membership

$\gamma_j(x_i) = P(z = j | X = x_i, \mu, \Sigma, \omega) = \frac{\omega_j \mathcal{N}(x_i, \mu_j, \Sigma_j)}{\sum_{l=1}^k \mathcal{N}(x_i, \mu_l, \Sigma_l)}$

- **E-step:** compute cluster membership weights $\forall$ pt using current params $\quad \gamma_j^{(t)}(x_i) = \frac{\omega_j^{(t-1)} \mathcal{N}(x_i; \mu_j^{(t-1)}, \Sigma_j^{(t-1)})}{\sum_{l=1}^k \mathcal{N}(x_i; \mu_l^{(t-1)}, \Sigma_l^{(t-1)})}$
- **M-step:** fit clusters to weighted pts $\quad \omega_j^{(t)} = \frac{1}{n} \sum_{i=1}^n \gamma_j^{(t)}(x_i)$ $\quad \mu_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)\, x_i}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$ (*)

5. TIME-SERIES MODELING

what if data is not iid but generated over time? temporal factor

• Markov models

given obs, what's next? $x_t \to x_{t+1} \to x_{t+2} \to x_{t+3}$

$x_t \in \{1 \dots c\}$ states $\quad t = 1 \dots T$

$P(Y_1 = q) = p_q \qquad P(Y_t = q | Y_{t-1} = q') = \theta_{q|q'}$ ⚠ no memory, only dep on previous state

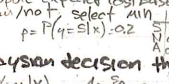
$P(Y_t | Y_{t-1}, Y_{t-2}, \dots, Y_1) = P(Y_t | Y_{t-1})$ $\forall t$

$P(Y_{t+1} | Y_t = q') = P(Y_t = q | Y_{t-1} = q') = \theta_{q|q'}$ $\forall q, q', t$ stationary assumption

k-th order markov models → introduce memory

$P(Y_{t+1} | Y_{t-k+1}, \dots, Y_t)$ $\quad$ k previously states

extend state Y_t to a multiple-dim state $Z_t = (Y_{t-k+1} \dots Y_t)$



Parameters $\theta_{x_{t+1}|x_t}$ depends on order k of chain, 1st order chain, c states → $c(c-1)$ $\quad$ $= c^2 - c$ & constraint $\sum_{j=1}^c \theta_{j|q} = 1$

in general $\quad \#\theta_{x_{t+1}|Y_{t-k+1:t}} \in O(c^{k+1})$

• Predictions

ex weather of tomorrow?

if we know initial & transition probabilities → easy

ex 2 timesteps $\quad P(Y_{t+2} = q | q_{1:t}) = \sum_{q'} P(Y_{t+2} = q, Y_{t+1} = q' | q_{1:t})$

$= \sum_{q'} P(Y_{t+2} = q | q_{1:t}, Y_{t+1} = q') \cdot P(Y_{t+1} = q' | q_{1:t})$

to

$P(Y_{t+2} = q | q_{1:t}) = \sum_{q'} \theta_{q|q'} \cdot \theta_{q'|q} \cdot \theta_{q|q}$

state-probvector $p^{(t)} = [p_1^{(t)} \dots p_c^{(t)}] \in \mathbb{R}^c$ $\quad$ $T \in \mathbb{R}^{c \times c}$

trans. matrix

$T \vec{q}_t = \vec{q}_{t+1}$ $\quad$ $p^{(t+1)} = p^{(t)} T$ $\quad$ $p^{(t+k)} = p^{(t)} T^k$

• Learning markov chain

m sequences (usually 1)

$D = \{(q_1^{(1)}, q_2^{(1)} \dots q_{n_1}^{(1)}) \dots (q_1^{(m)}, \dots q_{n_m}^{(m)})\}$ $\quad$ estimate p_q and $\theta_{q|q'}$

compute likelihood of data

$P(D|\theta) = \prod_{i=1}^m P(q_{1:n_i}^{(i)} | \theta) = \prod_{i=1}^m \dots = \prod_{i=1}^m p_{q_1^{(i)}} \prod_{t=2}^{n_i} \theta_{q_t^{(i)} | q_{t-1}^{(i)}}$ $\quad$ over m sequences

$\log P(D|\theta) = \sum_{j=1}^c \#(Y_1 = j) \log p_j + \sum_{j=1}^c \sum_{j'=1}^c \#(Y_{t+1} = q | Y_t = j') \log \theta_{j|j'}$

observations mapped to state based on frequency

MLE $\quad p_q^* = \frac{\#(Y_1 = q)}{m} \qquad \theta^*_{q|q'} = \frac{\#(Y_{t+1} = q | Y_t = q')}{\#(Y_t = q')}$

Contiguous seq. prediction

idea: sliding window size k, only use k most recent observ

④ #random/natural variables, remains the same $\forall t$

Gaussian linear time series

introduce restriction on prediction: $P(Y_{t+1} = q | q_{t-k+1:t}) = \mathcal{N}(q; \omega_0 + \sum_{i=1}^k \omega_i q_{t-k+i}, \sigma^2)$

Gaussian autoregressive model of order k $\quad$ weighted sum of previous observations

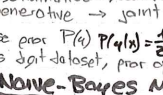
can reduce it to lin regr

ex k = 2

for $q_{t+1}, \mu = \omega_1 q_{t-1} + \omega_2 q_t$, var σ^2 → lin regr with quadratic loss fct

reinterpret original seq as set of feature vectors and labels

$D = \{(x_i', q_i')\}_{i=1 \dots t-2}$



compute w, σ^2 to max the (likelihood)

of the lin. regr model defined on q

$(w, \sigma^2) = \arg\max P(q_{3:t} | q_{1:2}, w, \sigma^2) = \prod_{i=3}^t P(q_i | q_{i-2}, q_{i-1}, w, \sigma^2) = \prod_{i=1}^{t-2} \mathcal{N}(q_i'; w^T x_i', \sigma^2)$

transformation

for general memory size k, $q_{1:t}$ is transformed into

Gaussian non-lin time series

$P(Y_{t+1} = q | q_{t-k+1} \dots q_t) = \mathcal{N}(q; f(q_{t-k+1:t}; \theta), \sigma^2)$

for some non-linear fct f

thing is like $\quad q_{t+1} = f(\dots) + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \sigma^2)$

$\nabla_w L(w) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(w) \\ \vdots \\ \frac{\partial}{\partial w_d} L(w) \end{bmatrix}$ $\quad$ $\nabla_w (L(w) + C(w)) = \nabla_w L(w) + \nabla_w C(w)$

EM

▷ stat. model generating X = observation, Z = latent variables (unknown)

θ params (unknown), likelihood fct $L(\theta; X, Z) = p(X, Z | \theta)$

→ MLE = $\arg\max_\theta$ of $\quad L(\theta; X) = p(X|\theta) = \int_Z p(X, Z|\theta)\, dZ$

EM: find MLE in 2 steps:

(E) $Q(\theta | \theta^{(t)}) = E_{Z|X,\theta^{(t)}}[\log L(\theta; X, Z)]$ ← est of log-likelihood under current estimate

(M) $\theta^{(t+1)} = \arg\max_\theta Q(\theta | \theta^{(t)})$ $\quad$ maximize over θ

ideally, want to max $\log P(X|\theta)$ (log-likelihood of observ), but since it's hard, ↑

$\max_\theta Q^{(t)}(\theta)$ ← the expected

Notation $\quad A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

MLE + prior = MAP

(*) $\quad \Sigma_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)(x_i - \mu_j^{(t)})(x_i - \mu_j^{(t)})^T}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$

$\frac{\partial}{\partial x} \ln(x) = \frac{1}{x}$

$Q^{(t)}(\theta) := E_{p(Z|X,\theta^{(t)})}[\log P(X,Z|\theta)]$

$p^{(t)}(z) := P[Z | x, \theta^{(t)}]$

$E_{Q(Z|X)}\left[\log P(x_{1:3}, z_{1:3})\right] = E_{Q(Z|X)}\left[\log \prod_{i=1}^3 P(x_i; z_i)\right] = \sum_{i=1}^3 E_{Q(Z|X)} \log P(x_i, z_i)$

$= \sum_{i=1}^3 \sum_{q=1}^3 Q(z=z_i | x=x_i) \log P(x_i, z_i) = \log P(z=z_i) P(x=x_i | z=z_i)$

$= \sum_{i=1}^3 Q_{i1} \log \underbrace{P(z=1)}_{\pi} \underbrace{P(x=x_i | z=1)}_{\text{Pois}(\lambda_1)} + Q_{i2} \log \underbrace{P(z=2)}_{1-\pi} \underbrace{P(x=x_i | z=2)}_{\text{Pois}(\lambda_2)}$ $\quad$ not ws

$= \sum_{i=1}^3 Q_{i1} (\log \pi + x_i \log \lambda_1 - \lambda_1 - \log(x_i!)) + Q_{i2} (\log(1-\pi) + x_i \log \lambda_2 - \lambda_2 - \log(x_i!))$

Pois: $P(X=x) = \frac{\lambda^x}{x!} e^{-\lambda}$ $\quad$ compare the log-likelihood of the models on the test-set ← primordial approach for select model

NBC: $P(x | q=k) = \frac{d}{i=1} \underbrace{P_{ki}^{(x_i)}}_{\text{if } x_i = 1} \underbrace{(1 - P_{ki})^{1-x_i}}_{\text{else}}$ $\quad$ $\theta = \begin{cases} p_{ki} & \text{if } x_i = 1 \\ 1 - p_{ki} & \text{else} \end{cases}$

not PSD $\quad \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \overset{\text{use}}{=} \begin{pmatrix} \sim & \sim \\ \sim & \sim \end{pmatrix}$

$\mathcal{N}$: $\phi(0, v, \omega) = \phi\left(\frac{0 - Y}{\sqrt{\omega}}, 0, 1\right)$

$\begin{pmatrix} 6 & 6 \\ 6 & 8 \end{pmatrix} \overset{\text{use}}{\to} \begin{pmatrix} \sim & \sim \\ \sim & \sim \end{pmatrix}$

K-means++ // target good initialization of centers

1. start with $\mu_1 = x_i$, ie u.a.r
2. add center randomly, $\mu_k = x_i$ s.t. prob $(i) = \frac{1}{Z} \min_{j=1 \dots k-1} \|x_i - \mu_j\|_2^2$ (Z is normalizing param)

↳ the further away, the more likely to select

Expect. = $O(\log k) \cdot$ (optimal k-means)